Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2278 – 2287

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

Relieving the Effects of Uncertainty in Forest Fire Spread Prediction by Hybrid MPI-OpenMP Parallel Strategies

Tomàs Artés[a,b,*], Andrés Cencerrado[a,c], Ana Cortés[a,d], Tomàs Margalef[a,e]

[a] *Computer Architecture and Operating Systems Department, Escola d'Enginyeria, Universitat Autònoma de Barcelona, Campus UAB, Bellaterra 08193, Spain*

[b] *tomas.artes@uab.cat*

[c] *andres.cencerrado@uab.cat*

[d] *ana.cortes@uab.cat*

[e] *tomas.margalef@uab.cat*

Abstract

The accurate prediction of forest fire propagation is a crucial issue to minimize its effects. Several models have been developed to determine the forest fire propagation. Simulators implementing such models require diverse input parameters to deliver predictions about fire propagation. However, the data describing the actual scenario where the fire is taking place are usually subject to high levels of uncertainty.

The input-data uncertainty represents a serious drawback for the correctness of the prediction. So, a two-stage methodology was developed to calibrate the input parameters in an adjustment stage so that the calibrated parameters are used in the prediction stage to improve the quality of the predictions. This way, we relieve the effects of such uncertainty. In this work, we take advantage of this two stage methodology applying Genetic Algorithms as the calibration technique. However, the use of Genetic Algorithms require the execution of many simulations. This fact, added to the eventual long executions of the underlying simulator (due to its inherent complexity), implies to deal with another serious problem: the time needed to deliver the predictions. To be useful, the prediction must be provided much faster than real time. So, it is necessary to exploit all available computing resources.

In this work, we present a two-stage forest fire spread prediction hybrid MPI-OpenMP application based on the Master-Worker paradigm and the parallelization of the FARSITE simulator in order to minimize the response time. The results as regards the enhancement in the quality of the predictions are reported, as well as the results regarding the time saving obtained by this hybrid application.

Keywords:
Input Data Uncertainty, High Performance Computing, Prediction Framework, OpenMP, MPI, Hybrid Applications, Evolutionary Computation, Forest Fires

1. Introduction

Natural hazards still represent a challenge for society. When these kinds of disasters occur it is crucial to organize a prompt response to minimize the damages. Therefore, some of these natural catastrophes have been modeled and simulators have been developed to predict their evolution [1][2]. Forest fire is one of these natural

*Corresponding author. Tel.: +34-93-581-1990 ; fax: +34-93-581-2478 .

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.399

phenomena which is common at some geographical areas such as southern Europe countries. This natural hazard involves important social and environmental impacts, such as possible loss of human life, infrastructure damages, social disruption and unnecessary pollution, among others. Moreover, the massive combustion of huge quantities of fuel is another important aspect to be considered because of the important waste of energy produced.

In order to tackle this problem, different physical models have been developed and implemented into forest fire spread simulators. The Rothermel model [3] can be considered the most used and representative. The FARSITE simulator [4] is an implementation of the Rothermel model. This simulator requires input parameters that in some cases are uniform on the whole terrain and constant in time, but some other parameters vary from one point of the terrain to another one or present a temporal evolution. Furthermore, there are serious difficulties to gather precise values of certain parameters at the right places where the catastrophe is taking place, often because the hazard itself distorts the measurements. So, in many cases the unique alternative consists of working with interpolated, outdated, or even absolutely unknown values. Obviously, this fact adds the problem of dealing with high levels of uncertainty in the input parameters, which results in a lack of accuracy and quality on the provided predictions.

To overcome the input data uncertainty, a two stage prediction methodology was developed [5]. This methodology divides the prediction process into two different stages: adjustment and prediction. During the first stage, a calibration method is applied using data about the observed fire propagation to obtain the input data set which best describes the actual evolution of the catastrophe. Afterwards, the prediction stage starts a new simulation using the values obtained in the calibration stage. So far, several adjustment techniques have been tested to calibrate the input-parameter set, from which stands out the use of evolutionary computation, specifically Genetic Algorithms (GAs) [6].

Nevertheless, in order to be useful, any evolution prediction of an ongoing hazard must be delivered as fast as possible in order not to be outdated. In the area of emergency management, the concepts of urgency and accuracy are closely related. The GA requires the execution of a certain number of iterations with several simulations per each iteration. Therefore, the GA has been implemented applying a Master/Worker paradigm to take advantage of parallel/distributed systems running different simulations of each iteration simultaneously. In certain cases, the inherent complexity of this simulator leads us to very long execution times. For this reason, we carried out the parallelization of the simulation kernel, taking advantage of the capabilities of multicore architectures.

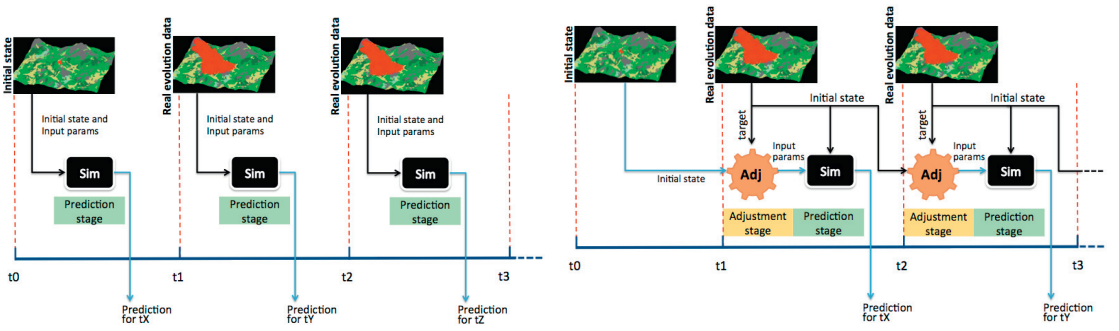
This way, a hybrid MPI-OpenMP application has been implemented in order to benefit from merging the two-stage method, the Master-Worker paradigm and the FARSITE simulator parallelization. Thus, we are able to shorten the overall prediction process and improve the accuracy of the predictions.

This work is organized as follows. In the subsequent Section 2 the two-stage methodology and its implementation is introduced. Afterwards, the simulation kernel is analyzed and the parallelization of the simulator is described in Section 3. The study and analysis of the achieved results by the hybrid MPI-OpenMP applicatin are reported in section 4. Finally, the main conclusions and open lines for future work are described in Section 5.

2. Two-Stage Prediction Scheme

The classical way of predicting forest fire behavior (see Figure 1(a)) takes the initial state of the fire front as input, as well as the input parameters given for a certain time instant. The simulator then returns the fire spread prediction for a later time instant.

Comparing the simulation result with the advanced fire line, the forecasted fire front tends to differ to a greater or lesser extent from the real fire line. The main reason of this mismatch is that the classic calculation of the simulated fire is based on one single set of input parameters afflicted with the aforementioned uncertainty. To overcome this drawback, a simulator independent data-driven prediction scheme was proposed to calibrate model input parameters [5]. Introducing a previous adjustment stage as shown in Figure 1(b), the set of input parameters is calibrated before every prediction step. Thus, the proposed solution comes from reversing the problem: how to find a parameter configuration such that, given this configuration as input, the fire simulator would produce predictions that match the actual fire behavior. Having detected the simulator input that better reproduces the observed fire propagation, the same set of parameters can also be used to describe the immediate future, assuming that meteorological conditions remain constant during the next prediction interval. Then, the prediction becomes the result of a series of automatically adjusted input configurations.



(a) Classic prediction (b) Two-stage prediction

Fig. 1. Prediction Methods

The two stage prediction method can be applied continuosly, providing calibrated parameters at different time intervals and taking advantage of observed fire behavior. Thus, this approach has been proven to be appropriate in order to enhance the quality of the predictions, relieving the disadvantages related to the input-data uncertainty.

Previous works, such as [7, 8, 9], proposed several calibration techniques, which made the problem of fire spread prediction fit the *Dynamic Data-Driven Application Systems* (DDDAS) paradigm [10, 11]. Because of its outstanding results, in this work we use the GAs-based adjustment technique.

GAs work in an iterative way. This algorithm starts with an initial population of individuals which will be evolved through several iterations in order to guide them to better search space areas. The individuals used in the case of forest fire spread prediction are defined as a sequence of different genes, namely wind speed and wind direction, moisture content of the live fuel, moisture content of the dead fuel (at three different times) and type of vegetation. Operators such as *elitism*, *selection*, *crossover* and *mutation* are applied to every population to obtain a new one superior to the previous one. As it is well known, the use of evolutionary techniques is highly computational-demanding, since it involves large sets of simulations. Due to the existing time constraints when dealing with an emergency, such as a forest fire, this fact represents an important inconvenience. In addition, as previously stated, the underlying simulator which is being used may present prohibitive execution times, depending on the scenario being simulated. Consequently, the use of the two-stage prediction framework to overcome the input-parameter uncertainty problem presents the drawback of the long time needed to deliver reliable predictions. For this reason, we have to rely on the so-called High Performance Computing (HPC) techniques to avoid the prediction time to last too long.

So, in order to compute the adjustment stage in a reasonable time interval, an MPI-based framework under the Master/Worker paradigm has been implemented. Thus, we can take advantage of the potential possibility to carry out different simulations of each generation in a parallel way on parallel/distributed systems.

Figure 2 summarizes how this framework is designed. In the first prediction stage, the Master node generates a population that represents a set of different scenarios to be evaluated (the initial population of individuals). Then, the Master node distributes these individuals among the Worker nodes. Each worker simulates the received individual, i.e. the scenario that represents the individual, and the output is compared to the actual fire behavior by the evaluation of the difference between the actual propagation and the simulated propagation. The difference is computed using equation 1 and it is used as error measure.

$$Difference = \frac{(UnionCells - InitCells) - (IntersectionCells - InitCells)}{RealCells - InitCells} \quad (1)$$

UnionCells is the number of cells which describe the surface burned counting simulated fire and the real fire. InitCells is the number of cells burned at the begining time in both maps, real and simulated. IntersectionCells is the number of cells burned together in the real map and also in the simulated map, while RealCells are the cells burned in the real map. This difference takes into account the wrong burned cells and the missing for burned cells.

The resulting error is sent back to the Master. When all the workers have finished, the Master node is able to rank the individuals in a list using the error as sorting criteria. Then, a new generation of individuals is created by applying the aforementioned genetic operations of elitism, crossover and mutation. Once the new population is generated, the new individuals are distributed again among the workers. This process is repeated in an iterative way a fixed number of generations, and the provided solution for the problem is the best individual of the last generation (the one which produces lowest error) is considered as the response of the adjustment stage. This individual (this set of input parameters describing the scenario) will be the individual used to simulate the fire spread for the next time interval.

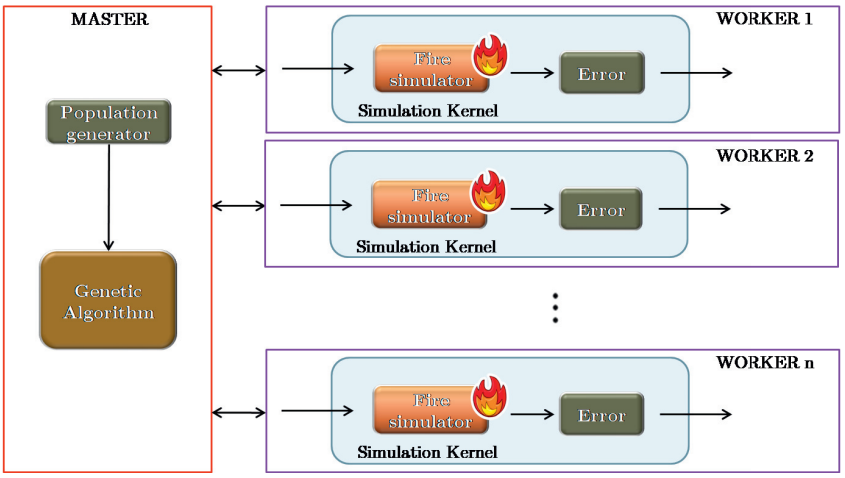


Fig. 2. GA implementation under MPI Master/Worker scheme.

As it can be seen, this scheme fits the well-known Master/Worker paradigm. At each generation, the Master node waits until all Workers compute the simulations and their respective errors. Therefore, the time incurred at each generation is defined by the worker which spents more time computing the individual provided by the Master. In the following Section, we deal with how to explote thre multicore capabilities of modern architectures in order to reduce the execution time of the most lengthy simulations and, therefore, reduce the time of each iteration of the GA.

3. Exploiting Multi-core Architecture

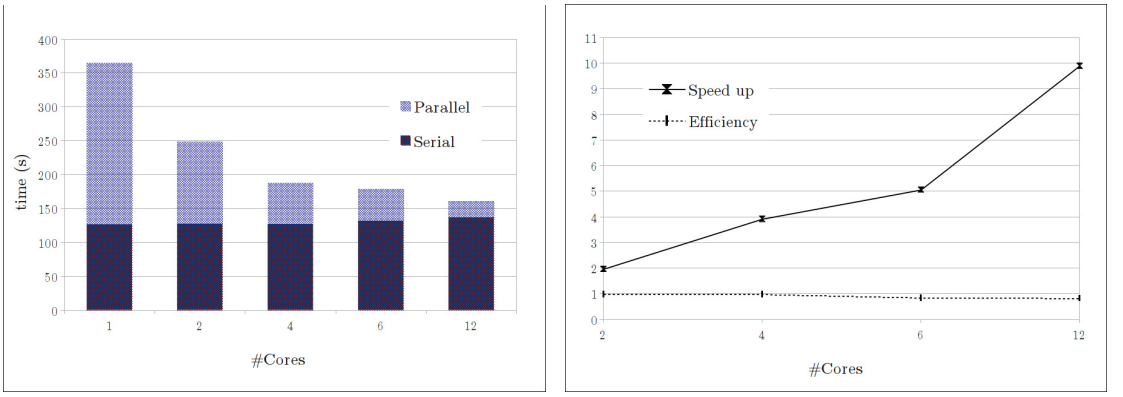
Considering that each individual (simulation) is allocated to different computational nodes, the time needed by a certain worker node depends on how long its specific simulation lasts. In previous studies [12], the significant execution time differences of FARSITE simulations has been analyzed. Depending on the different input data sets, the execution time of a single simulation may vary from few seconds to hours. Since the available FARSITE Linux version is a serial code, the release of a parallel version of this forest fire spread simulator would be an important improvement to reduce the total execution time exploiting the capability of the muticore nodes.

Taking into account that a single simulation may produce important repercussions in the time needed to perform a single evolution step (to carry out a single generation), a deep analysis of the FARSITE simulator was carried out in order to determine which loop of the FARSITE code spents more time. The main objective of this analysis was to exploit the existing (and widely used) multicore architectures to parallelize the simulator internal loops using shared memory OpenMP pragmas.

FARSITE has been analyzed using gprof [13] under different scenarios to find the most time consuming functions. This analysis shows that the exclusive execution time is quite distributed all over the functions except two functions: *Cross()* function was consuming over 21.86% and *GetPerimeter1Value()* around 14.88% of inclusive execution time. Focusing on the code of the *Cross()* function, we realize that it is not an easy parallelizable code with OpenMP due to possible race conditions caused by a considerable number of shared variables in the loop

as well as other inconveniences (such as loop breaking), which can be fixed using more shared variables and incrementing the racing conditions. On the other hand, *GetPerimeter1Value()* function gives no opportunity to use OpenMP. So, we search over the callgraph for other functions with parallelizable loops, which include the calls to those functions. We found a function, *CrossCompare()*, which includes over a 60% of inclusive execution time and it is located at an upper level in the callgraph but in the same branch where the heavy inclusive execution time functions are located. This function includes a loop that could be parallelized using OpenMP. This loop is in charge of performing the fire line coherency control. Since the fire line is expressed with polygons, a faster fire propagation implies bigger perimeters, and therefore slower executions.

Thus, FARSITE has been parallelized by introducing OpenMP pragmas in such function. Because of Amdahl's law, the theoretical maximum speedup is limited by the part of the application that has been parallelized. Since this loop represents about 60% of the total execution time, it means that 40% of the execution time corresponds to a sequential part and this time cannot be reduced by this parallelization. To test the correctness of this approach, a test fire with a representative workload has been executed using different number of cores (2, 4, 6, and 12). In Figures 3(a) and 3(b) it can be observed that the obtained results regarding execution time, speedup and efficiency are quite close to the expected ones. OmpP [14] profiling tool has been used to get serial and parallel region execution time. The serial part of FARSITE lasts around 140 seconds and the parallel part lasts almost 220 seconds using one single core. Using two cores the parallel part is reduced to nearly 120 seconds, using four cores it is reduced to 60 seconds and using 12 cores the parallel part lasts less than 20 seconds. The total FARSITE execution time is reduced from 360 seconds to less than 160 seconds. The speedup of the parallel part can be seen in Figure 3(b).



(a) Execution time which is done in paralel or serial

(b) speedup and efficiency of the parallel part

Fig. 3. Parallel FARSITE

The parallel OpenMP FARSITE has been validated with different large scale forest fire providing exactly the same output files that the original sequential version. The parallel OpenMP FARSITE version has been executed considering different fire conditions with 1 core and 4 cores with 32 nodes IBM x3550 cluster. Each computing node has two dual core Intel Xeon 5160 and 12 GB of memory at 667Mhz.

Figure 4 shows the percentage of execution time reduction when the same fire is executed on 4 cores. It can be observed that a reduction higher than 35% is achieved when serial time is over 30 seconds.

4. Evaluating hybrid MPI-OpenMP Forest Fire spread prediction application

The two stage methodology has been parallelized in a hybrid MPI-OpenMP application under a MPI Master/Worker paradigm with shared memory, and OpenMP parallelism exploited at the Worker level.

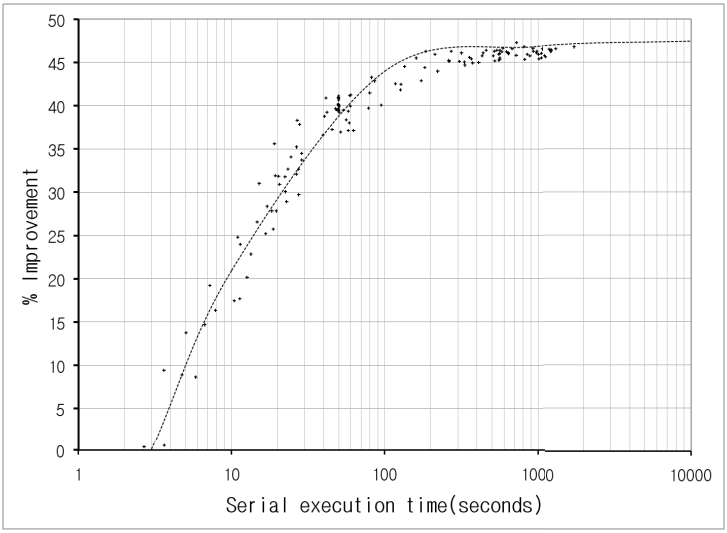


Fig. 4. Execution time improvement compared to serial time

In order to test this hybrid application on different scenarios, an experimental study has been carried out. This experimental study consists on 10 executions considering different initial populations and has been repeated for different reference fires. The terrain selected to run the experiment is the *Cap de Creus*, in the north-east of Catalonia (Spain). It has been selected due to the high number of fire occurrences on that area over the last 30 years. The meteorological data has been obtained from SMC (Catalan Meteorological Service) with XEMA (Automatic Weather Station Network) center and vegetation types has been obtained from CORINE land cover data base [15]. The total surface represented in the map is 1004304 cells (3012,912 ha).

Three different experiments have been carried out considering different computational resources and configurations:

- **Experiment A:** The first experiment consists of an execution of the GA using a population composed of 25 individuals, being evolved over 10 generations. The resources provided for this execution are 25 cores, each individual of the population being assigned to one core.
- **Experiment B:** The second experiment reproduces the same conditions, but in this case each worker is executed on 4 cores located in the same computing node. This second experiment has been executed using exactly the same populations that were generated at each generation of the first experiment by the GA in order to compare the execution time in the same situation.
- **Experiment C:** The first experiment and the second one should provide exactly the same error for each generation, but the second one should be faster since it is using 4 times more cores. So, a third experiment was established to compare the behavior with the same resources. The idea is to use 100 cores (25x4), dedicating each core to execute a different individual. It means that this time the population of the GA will be composed of 100 individuals instead of 25 individuals.

As first case of study, a reference fire with a simulation time over 20 seconds was considered. Figure 5 shows the obtained results. Each point in the depicted series represents the error obtained at the end of each generation. It can be observed that experiment B results in a good final error (around 15%) in less than 300 seconds, while the experiments A and C need almost 500 seconds to reach the same error. If we compare the results of experiment A with the ones of experiment B, it can be observed that the error corresponding to each generation of the GA is exactly the same since the populations are exactly the same, but the execution time is reduced about 40%. On the other side, the larger population of 100 individuals produce a better initial error, but afterwards the convergence is slower, considering the elapsed times. Each iteration is slower than in experiment B, as well. Based on these results, it can be settled that using more cores per individual improves the execution time of the adjustment stage.

As previously mentioned, the execution time of each individual depends on the particular values of the input parameters. Some individuals run in few seconds, whereas others may take minutes. So, the same experiments have been repeated using as the reference fire a fire with longer simulation time (around 120 seconds). Figure 6 shows the results obtained. In this case, the results of experiment B reach 15% error in 1100 seconds, the evolution process in experiment C needs 1500 seconds and the configuration of experiment A needs more than 2000 seconds. It can be observed that the use of 4 cores provides a larger improvement in execution time compared with experiment A configuration. This fact is consistent with the experiments related in Figure 4. On the other hand, the experiment C takes more than 3000 seconds to execute 10 iterations, whereas the experiment B configuration just takes 1350 seconds. This fact is due to the appearance of individuals lasting very long in particular iterations which are executed serially.

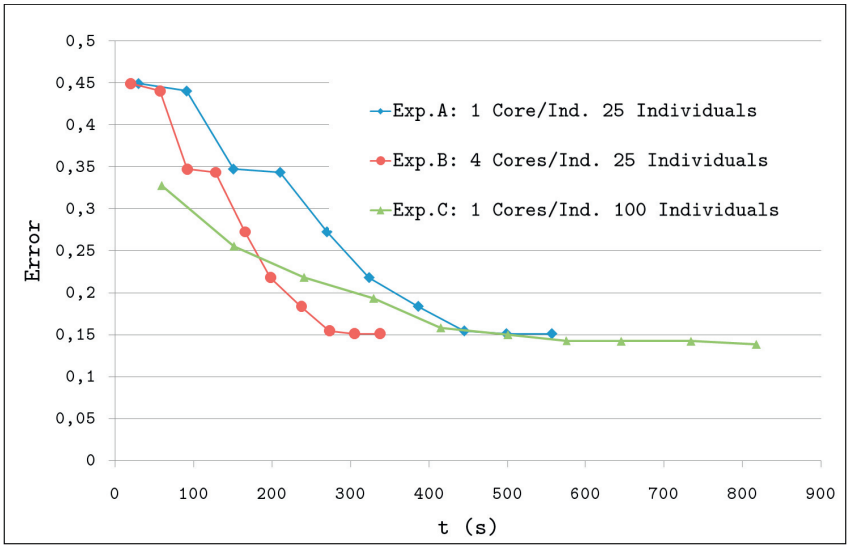


Fig. 5. Average error obtained in experiments A, B and C (Equation 1). Reference fire around 20 seconds of simulation time.

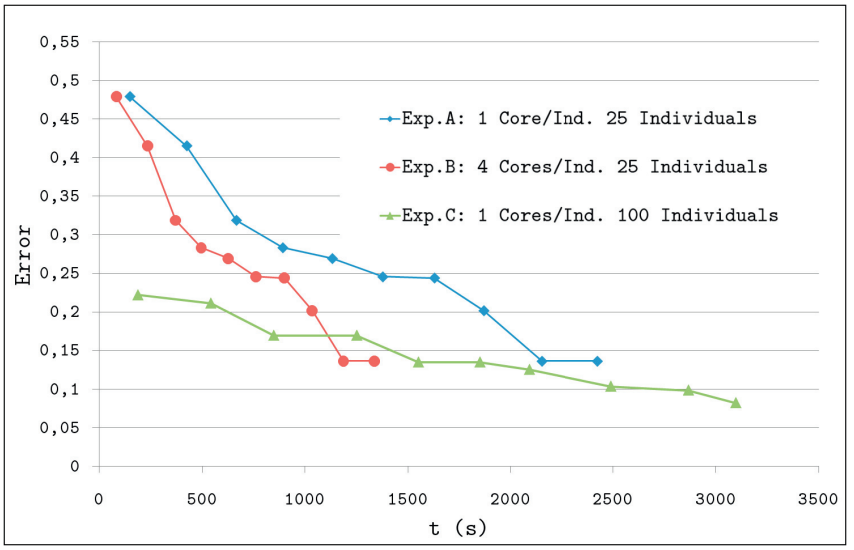


Fig. 6. Average error obtained in experiments A, B and C (Equation 1). Reference fire about 120 seconds of simulation time.

4.1. Resource Allocation to Speedup the Adjustment Stage

From the previous experimentation it can be deduced that the multicore parallelization of FARSITE is a very promising approach since it reduces the time to adjust the input parameters and therefore it reduces the whole prediction time. However, since the execution time of FARSITE depends on input parameters and the execution time of one iteration of the GA depends on the largest individual of the population, it would be possible to reduce the adjustment time by dedicating more computing resources (cores) to the longest individuals of each population and less computing resources to fast individuals, in such a way that the load among the workers is balanced by allocating more cores to longer individuals. So, there arises the need to predict the simulator execution time depending on the input parameters to determine how many cores should be allocated to that particular individual.

In [16], a methodology for characterizing the simulation kernel has been reported. Such methodology is based on Decision Trees [17] as a classification method and allows to estimate, in advance, the execution time of a certain simulator execution, given a new set of input parameters.

Figure 7 depicts the histogram of the simulation times of FARSITE in a testbed of 12000 different instances of the single-core version. It can be observed that many simulations lasts for less than 100 seconds while others lasts more that 3000 seconds. This histogram has been divided in 4 classes and the decision tree is able to classify a new individual in the appropriate class. So, it is possible to have a boundary of the execution time of each individual.

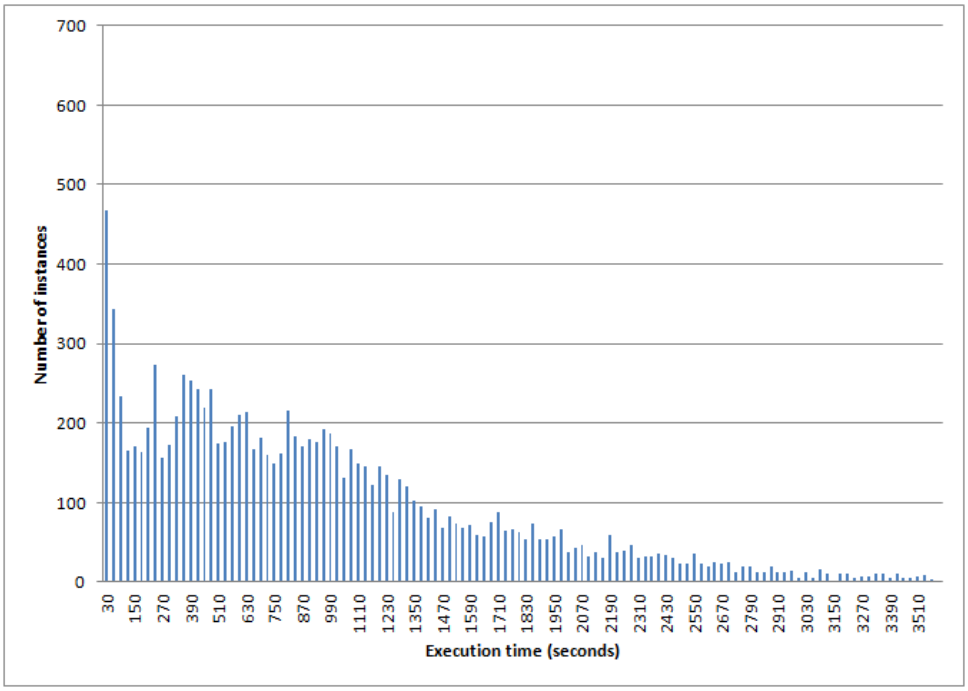


Fig. 7. Single-core version

The same experiment has been executed considering 4 cores per individual. Figure 8 depicts the histogram of the simulation times of FARSITE in the same testbed running on the multicore version using 4 cores. As one can notice, the histogram of the multicore version can be viewed as a *compressed* histogram of the single-core version. In this case most individuals lasts less than 200 seconds and only a few of them lasts more than 1000 seconds.

By means of the classification performed by the Decision Trees, we are able to detect the longest simulations in the evolutive process. Moreover, we are able to estimate the execution time they will produce. Taking into account the speedup produced by the multicore version, we can take advantage of this fact and design proper policies in order to exploit the available computational resources, and to shorten the overall adjustment and prediction processes. This would represent a great benefit in the presence of strict time constraints, as in the case of managing a forest fire.

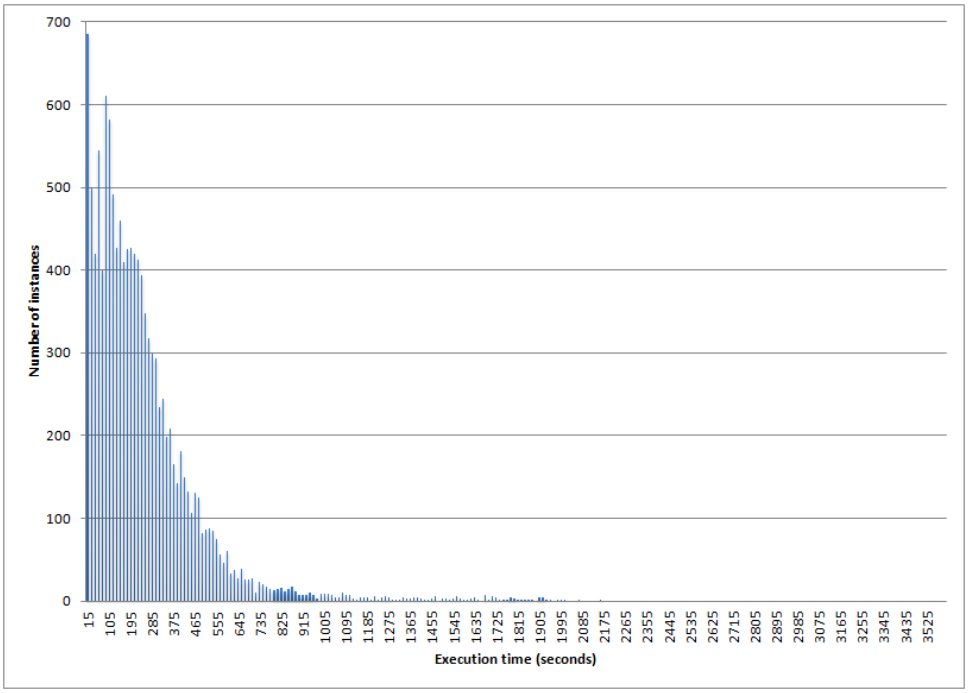


Fig. 8. Multicore version (4 cores)

Currently, we are focusing on that and, for this purpose, a wide statistical analysis is being carried out, designing different node-allocation policies according to the existing constraints and computational nodes availability.

5. Conclusions and Future Work

Forest fire spread prediction is undoubtedly a relevant application area in which the Computational Science can play a very important role. In fact, since many years ago, many research efforts have been focused on the implementation of different models in order to better predict how an ongoing fire will behave in the near future.

Nevertheless, this hazard represents very difficult system to simulate. Theoretical and model-related issues aside, many simulators lack precision on their results because of the inherent uncertainty of the data needed to define the state of the system. This uncertainty is due to the difficulty to gather precise values at the right places where the catastrophe is taking place, or because the hazard itself distorts the measurements. So, in many cases the unique alternative consists of working with interpolated, outdated, or even absolutely unknown values. Obviously, this fact results in a lack of accuracy and quality on the provided predictions.

As a suitable method to overcome this drawback, a two-stage prediction method was presented. This method relies on the adjustment of the simulator input parameters before carrying out the prediction for a latter time instant. It has been demonstrated that this method highly improves the accuracy of the results. However, it could involve a serious increase on the time needed for the prediction process.

So, for this strategy to be useful, it is compulsory to design and develop methods which allows us to minimize the time incurred in carrying it out. For this purpose, we present a hybrid MPI-OpenMP platform which takes advantage of the potential of parallelization of both the adjustment technique and the underlying simulator.

The results obtained from our testbeds highlight the great benefit we can obtain from the application of this framework, presenting a significant decrease in the overall execution time, keeping intact the gain obtained by applying the two-stage prediction method, as regards the quality of the predictions.

Furthermore, these results allow us to set out further optimizations. The fact of being able to estimate, in advance, how long a simulation will last, depending on the number of computational cores we can use at each

computational node emphasizes the benefit of the developed parallelization of FARSITE as the underlying simulator.

Thanks to this capability, we are able to tackle, with a guaranteed background, the problem of designing policies for the efficient allocation of different parallel versions of the simulator to different nodes, according to their multicore architecture features.

Finally, the MPI-OpenMP hybrid structure allows good results under an approach of time and precision that may enable the application of the methodology based on two stages using a genetic algorithm.

Acknowledgements

This research has been supported by MICINN-Spain under contract TIN2011-28689-C02-01.

References

[1] V. V. Titov, F. I. Gonzalez, P. M. E. Laboratory, Implementation and testing of the Method of Splitting Tsunami (MOST) model, Vol. 108, U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, Environmental Research Laboratories, Pacific Marine Environmental Laboratory ;Springfield, VA, Seattle, Wash., 1997.
URL http://purl.access.gpo.gov/GPO/LPS46461

[2] P. Vickery, J. Lin, P. Skerlj, L. Twisdale, K. Huang, Hazus-mh hurricane model methodology. i: Hurricane hazard, terrain, and wind load modeling, Natural Hazards Review 7 (2) (2006) 82–93. doi:10.1061/(ASCE)1527-6988(2006)7:2(82).

[3] R. C. Rothermel, A mathematical model for predicting fire spread in wildland fuels, Director (INT-115) (1972) 40.
URL http://www.srs.fs.usda.gov/pubs/32533

[4] M. A. Finney, FARSITE, Fire Area Simulator–model development and evaluation, Res. Pap. RMRS-RP-4, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, 1998.

[5] B. Abdalhaq, A. Cortés, T. Margalef, E. Luque, Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques, Future Generation Computer Systems 21 (1) (2005) 61 – 67. doi:10.1016/j.future.2004.09.013.
URL http://www.sciencedirect.com/science/article/pii/S0167739X04001360

[6] M. Denham, K. Wendt, G. Bianchini, A. Cortés, T. Margalef, Dynamic data-driven genetic algorithm for forest fire spread prediction, Journal of Computational Science 3 (5) (2012) 398 – 404, advanced Computing Solutions for Health Care and Medicine. doi:10.1016/j.jocs.2012.06.002.
URL http://www.sciencedirect.com/science/article/pii/S1877750312000658

[7] M. Denham, A. Cortés, T. Margalef, Computational steering strategy to calibrate input variables in a dynamic data driven genetic algorithm for forest fire spread prediction, in: Lecture Notes in Computer Science, Vol. 5545(2), 2009, pp. 479–488.

[8] G. Bianchini, M. Denham, A. Cortés, T. Margalef, E. Luque, Wildland fire growth prediction method based on multiple overlapping solution, Journal of Computational Science 1(4) (2010) 229–237.

[9] R. Rodríguez, A. Cortés, T. Margalef, Towards policies for data insertion in dynamic data driven application systems: a case study sudden changes in wildland fire, Procedia Computer Science 1 (1) (2010) 1267–1276.

[10] F. Darema, Dynamic data driven applications systems: A new paradigm for application simulations and measurements, in: Lecture Notes in Computer Science, Vol. 3038, 2004, pp. 662–669.

[11] F. Darema, Grid computing and beyond: The context of dynamic data driven applications systems, in: Proceedings of the IEEE, Vol. 93(3), 2005, pp. 692–697.

[12] A. Cencerrado, A. Cortés, T. Margalef, Genetic algorithm characterization for the quality assessment of forest fire spread prediction, Procedia Computer Science 9 (0) (2012) 312–320, proceedings of the International Conference on Computational Science, ICCS 2012. doi:10.1016/j.procs.2012.04.033.
URL http://www.sciencedirect.com/science/article/pii/S1877050912001548

[13] S. L. Graham, P. B. Kessler, M. K. McKusick, gprof: a call graph execution profiler, SIGPLAN Not. 39 (4) (2004) 49–57.

[14] K. Fürlinger, M. Gerndt, A Profiling Tool for OpenMP, OpenMP Shared Memory Parallel Programming (2008) 15–23.

[15] M. Bossard, J. Feranec, J. Otahel, CORINE land cover technical guide Addendum 2000, Technical report No 40, European Environment Agency, Kongens Nytorv 6, DK-1050 Copenhagen K, Denmark, 2000.
URL http://www.eea.europa.eu/publications/tech40add

[16] A. Cencerrado, A. Cortés, T. Margalef, On the way of applying urgent computing solutions to forest fire propagation prediction, Procedia Computer Science 9 (2012) 1657–1666.

[17] T. Mitchell, Machine Learning, McGraw-Hill, 1997.